ACTC/Research Performance Tools for Scientific Programming Current Status and On-going Work

David Klepacki, Luiz Derose,
Anshul Gupta
Advanced Computing Technology
IBM T.J. Watson Research

klepacki@us.ibm.com

IBM
Thomas J. Watson Research Center
PO Box 218
Yorktown Heights, NY 10598

ACTC Tools

- Goal
  - Complement IBM application performance tools offerings
- Methodology
  - Development in-house and in collaboration with universities, research labs, and industry
  - Deployment “AS IS” to make it rapidly available to the user community for use and evaluation
  - Close interaction between tools builders and application developers for quick feedback for functionality enhancements
- Disclaimer:
  - Member of IBM Research
  - Many designs invented in IBM Research
  - Not all became product

Disclaimer

Member of IBM Research

Many designs invented in IBM Research

Not all became product

David Klepacki klepacki@us.ibm.com

Current Collaboration Efforts

- **CEPBA (UPC) - Jesus Labarta**
  - Performance analysis of OpenMP programs
  - UTE / DPCL / Paraver interface
- **U. Illinois - Dan Reed, David Padua, & Josep Torrellas**
  - Pablo performance analysis and visualization system & I/O
  - Compiler support for data distribution and locality
- **U. Maryland - Jeff Hollingsworth**
  - Dyninst on AIX
  - Sigma & dynamic system for performance analysis
- **U. Wisconsin - Bart miller**
  - Paradyn on AIX
  - Tools scalability on large parallel systems

David Klepacki klepacki@us.ibm.com

Outline

- HPM Toolkit
  - Status and on-going work
- DPCL Tools
  - CEPBA Collaboration
- UTE Tools
  - UTE Gantt Chart
- SiGMA
- WSMP
  - Watson Sparse Matrix Package

David Klepacki klepacki@us.ibm.com

PMAPI (AIX 5)

- Performance monitor API (PMAPI) - developed by the IBM server division at the Austin lab
- Provide performance monitor data access at various levels
- Extension of the regular thread context

David Klepacki klepacki@us.ibm.com

HPM Toolkit Goals

- Correlate parallel source code with hardware and software performance measurements
- Help to identify causes of performance problems
- Low instrumentation overhead
- Flexible and easy to use interface
- Multi-language support
- Portability (AIX & Linux)

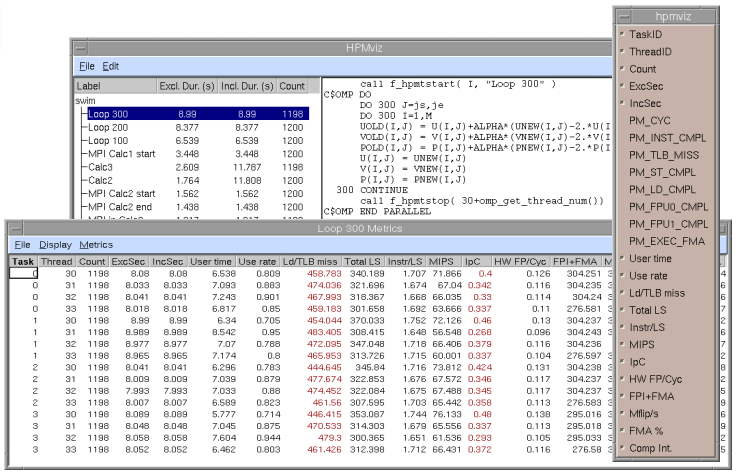
David Klepacki klepacki@us.ibm.com

HPM Toolkit Components

- hpmcount
  - Starts a program and at the end of the execution provides a summary output with:
    - Wall clock time
    - Resource utilization statistics
    - Hardware performance counters information
    - Derived hardware metrics
- libhpm
  - Instrumentation library for performance measurement of Fortran, C, and C++ applications
- hpmviz
  - Graphical user interface for visualization of hardware performance data

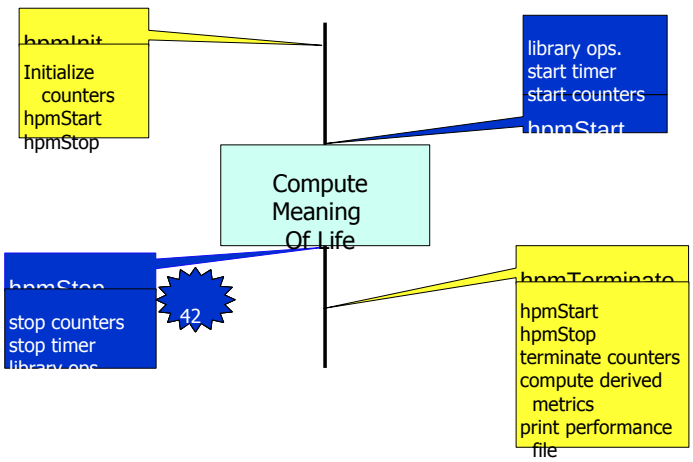
David Klepacki klepacki@us.ibm.com

HPMVIZ



David Klepacki klepacki@us.ibm.com

Measurement Process



David Klepacki klepacki@us.ibm.com

Current Status

- Downloadable via http://alphaworks.ibm.com
- Power 4 and AIX 5.1 support
- Automatic hardware counters profiler
  - Function level (loop level with OpenMP)
  - Binary instrumentation with DPCL
  - Functionality to filter or to select functions
  - Call graph generated during instrumentation
- New Visualizer
  - Call graph visualization support
  - Functionality to combine metrics from different runs

David Klepacki klepacki@us.ibm.com

AUTOHPM

- Automatic HPM instrumentation
- Simple interface
  - > autohpm a.out
- Support
  - serial
  - threaded
  - MPI
  - mixed applications
- Instrumentation
  - Functions
  - Loops (OpenMP)

David Klepacki klepacki@us.ibm.com

ACTC - CEPBA Collaboration

- CEPBA performance tools
  - Tracing
    - OMPtrace and OMPItrace
  - Visualization and analysis
    - Paraver

David Klepacki klepacki@us.ibm.com

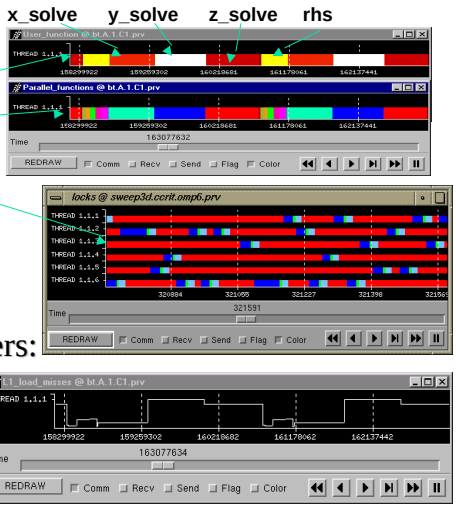
OMPtrace and OMPItrace

- OpenMP and MPI instrumentation
- Interfaces with
  - DPCL
  - PMAPI
- Automatic instrumentation
  - Entry and exit of user functions with parallel loops + user specified functions
  - Parallel functions generated by the compiler to encapsulate loop body
  - MPI calls

David Klepacki klepacki@us.ibm.com

Program Analysis - Paraver

- Basics:
  - Functions
  - Loops
  - Locks
- Hardware counters:
  - TLB misses
  - L1 misses
  - …



David Klepacki klepacki@us.ibm.com

PE Benchmarker

- Developed by the Performance Tools group at the IBM Poughkeepsie Lab
- Performance Collection Tool
  - MPI events
  - User defined events
  - Hardware performance counters
- Performance Visualization Tool

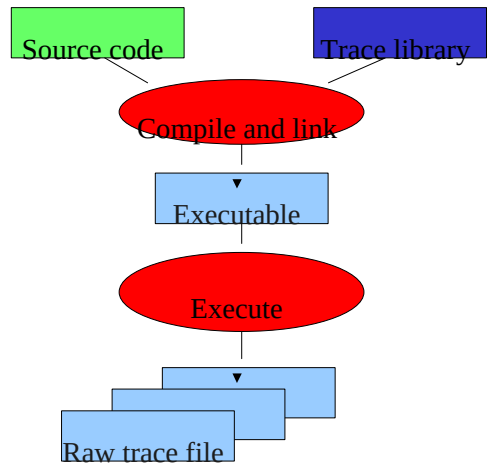
David Klepacki klepacki@us.ibm.com

Unified Trace Environment - UTE

- Provides a unified and easily expandable trace environment for MPI and other software layers on IBM SP systems
- Uses AIX trace facility
- Uses standard PMPI profiling interface
- Trace files generated independently
- Generates periodic global clock record
- Provides routines to generate user-defined events

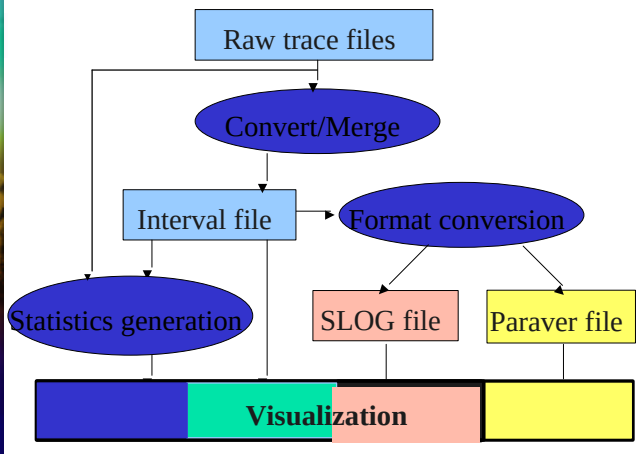
David Klepacki klepacki@us.ibm.com

Trace Generation Steps



David Klepacki klepacki@us.ibm.com

Post-Processing Steps



David Klepacki klepacki@us.ibm.com

UTE Visualization Tools

- Statistical viewer
- Jumpshot
- UTE Gantt chart
- Paraver

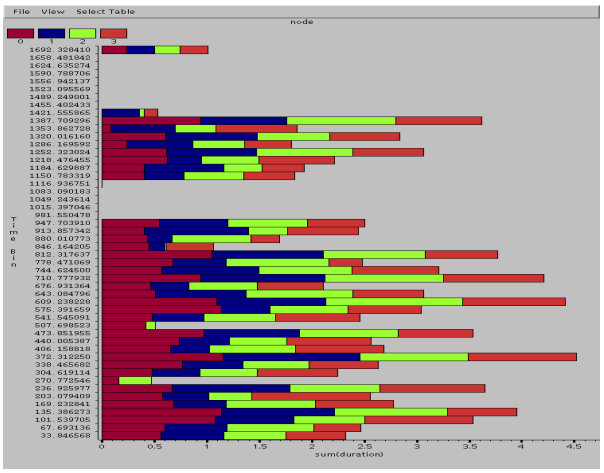
David Klepacki klepacki@us.ibm.com

Statistics Generation Utility

- Reads individual UTE interval files or merged ones
- A declarative language. Example: table name=sample condition=(start < 2) x=(“node”, node) x=(“processor”, cpu) y=(“avg(duration)”, dura, avg)
- Field names (start, node, cpu, duration) are defined in the profile
- Pre-defined statistics

David Klepacki klepacki@us.ibm.com

Statistics Viewer



David Klepacki klepacki@us.ibm.com

UTE Gantt Chart

- Reads individual or merged files
- Functions: zoom in/out, hide/show
- Integrated views: many in one
  - Disconnected-state thread view
  - Nested-state thread view
  - CPU (processor) view
  - Thread-processor view
- Call sequence viewer
- Source code browser

David Klepacki klepacki@us.ibm.com

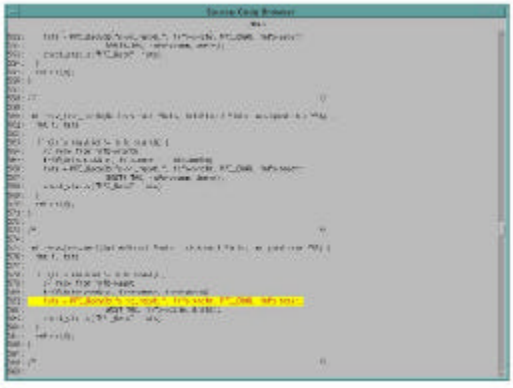
UTE Gantt Chart

•Thread View

•Processor View

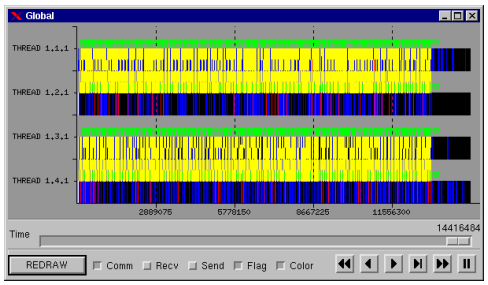
•Call sequence view

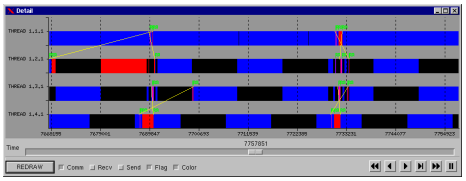
•Source code browser



David Klepacki klepacki@us.ibm.com

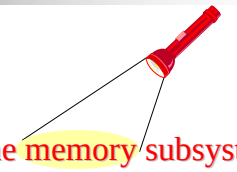
UTE - Paraver Interface





David Klepacki klepacki@us.ibm.com

Simulator Guided Memory Analyzer



- Where is the memory?
- Motivation
  - Efficient utilization of the memory subsystem is a critical factor to obtain performance on current and future architectures
- Goals:
  - Identify bottlenecks, problems, and inefficiencies in a program due to the memory hierarchy
  - Support serial and parallel applications
  - Support present and future architectures

David Klepacki klepacki@us.ibm.com

SiGMA Overview

- Family of tools to understand the memory subsystem
  - Focus on detailed statistics
  - Complement existing hardware counters
- Ability to handle real applications
  - Fortran and C
  - Serial and parallel programs
- Provide hints about restructuring
  - Padding (both inter and intra data structures)
  - Loop restructuring
  - Blocking parameters

David Klepacki klepacki@us.ibm.com

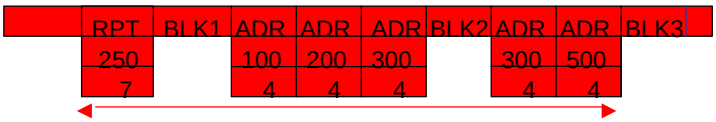
Approach

- Instrument program automatically
- Run instrumented program
  - Capture full information about memory use
  - Produce compact trace
    - Extracts loops and memory strides
- Post execution tools

David Klepacki klepacki@us.ibm.com

Representing Program Execution

- Capture full execution behavior
  - Record all basic blocks and memory addresses
  - Produces large traces (due to looping)
- Trace compression
  - Maintain pattern buffer
  - Scan for repeating patterns
    - Extract memory strides
  - Repeat algorithm for nested loops



David Klepacki klepacki@us.ibm.com

Post Execution Tools

- Memory profiler
  - Convert memory references back to array references
  - Share of accesses due to each data structure
  - Capacity, cold, and conflict misses can be identified
- Graphical user interface
  - Trace, source code, and data structure correlation
- Cache prediction tool
  - Predict cache misses using symbolic equations
- Detailed simulators
  - Cache simulator
  - TLB simulator

David Klepacki klepacki@us.ibm.com

Sigma Memory Dump

- Takes as input a trace file and the specifications of the memory subsystem.
  - Supports write-back, write-through caches, and the hardware prefetcher.
- Runs the cache simulation and dumps data into a 4D repository containing:
  - Control-flow (modules, functions, code segments)
  - Data structures
  - Metrics (LoadMisses, LoadHits etc)
  - Cache levels (L1, L2, TLB ...)

David Klepacki klepacki@us.ibm.com